2003

A Literature Review on Learner Control Strategies in Software Tutoring Systems

Xin Li
University of Nebraska at Kearney

Leen-Kiat Soh
University of Nebraska, lsoh2@unl.edu

University of Nebraska-Lincoln, Computer Science and Engineering
Technical Report # TR-UNL-CSE-2003-0006

A Literature Review on Learner Control Strategies in Software Tutoring Systems

Xin Li and Leen-Kiat Soh

Department of Computer Science and Engineering

University of Nebraska-Lincoln

June 2, 2003

Abstract

This paper is a comprehensive research review on the learner control strategies in software tutoring systems. With the application of more computer techniques in education and the involvement of more adults in software tutoring systems, the learner control strategy has become more appreciated than tutor control or program control. In this paper, the efficiency and necessity of learner control in software tutoring systems is discussed through the description of learning mechanism. Some typical applications of learner control strategies in software tutoring systems are presented. The available learner control strategies are classified from two perspectives: educational theory, and software mechanism. Finally, a set of open problems related with learner control strategies are raised and the possible solutions are proposed correspondingly.

Keywords: learner control, software tutoring systems, educational theory, CAI, ITS

1. Introduction

With the emergence of computers in the past decades, computer technologies have been significantly changing the content and practice of education. The consequent applications of multimedia, simulation, computer-mediated communication and communities, and internet-based support for individual and distance learning all have the potential for revolutionary improvements in education [McArthur *et al*. 1993, Forbus and Feltovich 2001]. The affordable, individualized teaching and learning environments become possible for a large and diverse population of students under real-world constraints such as limited financial resources and insufficient numbers of qualified instructors. The concept of education has been far more than traditional school education and the population of students is being broadened from children to nearly all adults for the needs of lifelong learning.

Tutoring and learning are two dependent aspects in education. The goal of tutoring is to help learners acquire more knowledge as well as improve their learning abilities. Thus, the development of tutoring patterns serves learning needs eventually. In history, tutoring patterns have evolved from the traditional one-on-one method and classroom method to the modern *computer-assisted instruction* (CAI) and *intelligent tutoring systems* (ITS). Correspondingly, learning patterns have also developed from the traditional pure textbook and classroom learning to *multimedia learning*, *distance learning*, as well as *interactive learning environments* (ILEs). CAI and ITS originate from the application of computer software and artificial intelligence (especially for the latter) to education, so the corresponding tutoring systems can be summarized as software tutoring systems. Similarly, the non-traditional learning can be summarized as *e-learning* [Clark and Mayer 2002], which is implemented on computers (via CD-ROM, Internet, or Intranet) and is designed to support individual learning or organizational performance goals.

Software tutoring system is a contrary concept to the traditional human tutoring system. The course tutor (not a human tutor) in a software tutoring system cannot enjoy the powerful face-to-face communication channels with students available in a classroom setting. Thus the control over learners is relatively weaker than in the traditional tutoring, where it is the tutor who is charge of the contents, sequence and pace of instruction and learning. In the software tutoring systems, however, it is easy for a learner to lose learning interest and leave the computers without the consent from tutors. Therefore, in order to obtain better tutoring outcomes, a software tutoring system should emphasize engaging students in the learning process and be adaptive to each individual learner.

The goal of the early software tutoring systems was to build clever tutors able to communicate knowledge to the individual learners. Recent and emerging work focuses on the *learner control* over the learning process such as learner exploring, designing, constructing, and using adaptive systems as tools [Kay 2001]. The concept of learner control is relative to the terms of tutor control, system control, and program control. When learners have the option to select the topics they want, control the sequence and pace at which they progress, and decide whether and how to use system-provided tools, the tutoring system can be said to offer learner control.

Learner control in tutoring systems is intuitively appealing, which embodies the subject and the objective of tutoring and learning. It can alleviate boredom, frustration, and anxiety of the learners because it enables learners to skip over materials they have previously learned or avoid materials they are not prepared to study [Steinberg 1977, Steinberg 1989]. It provides learners more autonomy. Although the benefits of learner control have not been explored thoroughly, the critical problem is no longer *whether* learner control should be applied to software tutoring systems but *how* to apply learner control strategies into software tutoring systems to maximize the learning benefits.

In this paper, the importance and necessity of applying learner control strategies into software tutoring systems is discussed. Some specific applications are also described to demonstrate the use of learner control strategies in software tutoring systems. A number of typical learner control strategies are presented from different perspectives. This paper also discusses some open problems and proposes corresponding solutions.

2. Learner Control

As early as 1957, Newman presented one of the first clear examples of learner control of instruction in the educational literature [Newman 1957]. But until 1961, the term of "learner control" was formally created and used in education by Mager [Mager 1961, Niemiec *et al*. 1996]. From then on, learner control is an important feature of most software tutoring systems.

2.1. Learning

The objective of applying learner control into software tutoring systems is to improve the efficiency of student learning. The learner control strategies should be built on types of learning and specific to the human being's learning mechanism.

2.1.1. Types of Learning

During the past one hundred years, three types of learning have evolved [Clark and Mayer 2002], and each of them can be seen in courses available in software tutoring systems: (1) *receptive learning*: information acquisition, used in lessons including lots of information with limited practice opportunities in order to satisfy "inform" goals, (2) *directive learning*: response strengthening, used in courses that require frequent responses from learners with immediate feedback in order to satisfy "perform-procedure" goals, and (3) *guided discovery*: knowledge construction, used in lessons that provide specific problems and supporting resources in order to satisfy "perform-principle" goals. These three learning types reflect a learning procedure from knowledge acquisition and review to the improvement of learners' problem solving abilities. In a specific course, they can be used in different lessons oriented to different learning phases.

2.1.2. Human Being's Learning Mechanism

Cognitive learning theory explains how mental processes transform information received by the sensory organs into knowledge and skills in human memory. In general, the human being has a three-level memory structure: sensory memory, working memory, and long-term memory. Cognitive learning theory explains the human being's cognitive process of learning with several key ideas [Clark and Mayer 2002]: (1) human memory has two channels, visual and auditory, and a limited capacity for processing information, (2) learning occurs by active processing in the memory system, and (3) new knowledge and skills must be retrieved from long-term memory for transfer to the job. Figure 1 illustrates the cognitive process involved in learning in software tutoring systems [Clark and Mayer 2002].

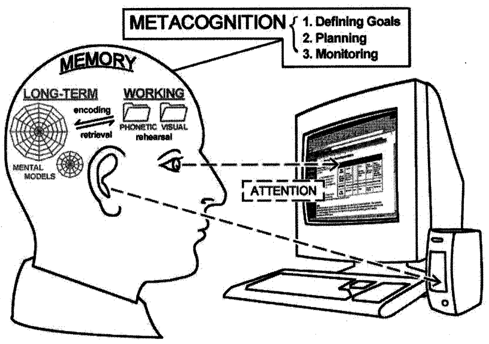

Figure 1. Cognitive Process of Learning

As shown in Figure 1, when a learner interacts with a software tutoring system, a lesson's visual and auditory information enters the learner's eyes and ears. This information is briefly stored in the sensory memory. It then enters working memory and is finally stored in long-term memory. The working memory is the center of cognition where all active thinking takes place but its capacity is very limited. Learning requires new knowledge and skills stored in working memory to be integrated with existing knowledge in long-term memory. The integration process is an *encoding* procedure, which requires active processing of the information in working memory. The active processing in working memory is called *rehearsal*. Later, the learner must be able to *retrieve* those skills from long-term memory back into working memory for solving problems.

It is obvious that the human being's learning is an active information processing procedure. Thus, it is very natural to use learner control strategies to optimize the learning outcome and increase the involvement and autonomy of learners in learning.

2.1.3. Learning Processes and Requirements

Due to the human being's learning mechanism, learning in software tutoring systems relies on four key processes [Clark and Mayer 2002]:

- The learner must focus on key graphics or words in the lesson to select what will be processed.
- The learner must rehearse the information in working memory to organize and integrate it with existing knowledge in long-term memory.
- New knowledge stored in long-term memory must be retrieved back on the job.
- Metacognitive skills manage and adjust these processes.

Because the learner's cognitive system has limited capacity competed by many sources of information, the learner must select those that best match his/her goals. Thus the software tutoring systems should provide learners with information selection opportunities as well as selection guidance. The integration work requires that the limited working memory capacity be not overloaded, so the systems should enable learners to reduce their cognitive loads. To support the knowledge retrieval process, the systems must provide a job context during learning that will contain the needed retrieval links for learners' use. In addition, just as a computer has an operating system to manage data transfer, the human processor has metacognition to manage the learning processes. *Metacognition* refers to the mental management processes that monitor those information processing. A learner with effective metacognitive skills is able to set learning goals, decide in effective ways to reach those goals, oversee the progress, and make necessary adjustments. A good software tutoring system should provide some of the management processes to enable high metacognitive learners to take advantage of them and also help learners with low metacognitive skills for successful learning.

2.2. Learner Control vs. Program Control

Learner control is a feature of tutoring systems where the learner can direct the tutoring flow provided by the system, thus guiding the system to respond to his/her own needs and interests [Steinberg 1977, Duchastel 1986, Clark and Mayer 2002]. Learner control strategy (LC) is contrasted with program control strategy (PC, also called system control or software control). LC allows each learner to control content sequencing and pacing, select learning contents, learning examples, the amount of practice exercises, and so forth, while PC lets a computer program make such decisions for each learner in a lockstep manner. LC and PC may coexist in a tutoring system but more LC means less PC and vice versa.

It is commonly assumed that LC functions as a means of accommodating individual differences of learners. Many studies have concluded that learners are the best judges of their learning needs and performance strategies. Some researchers comparing LC with the traditional PC reported the superiority of the former over the latter (e.g., [Park and Tennyson 1980, Ross *et al.* 1980, Gray 1987]), while other studies demonstrated the superiority of PC strategies (e.g., adaptive PC strategy with elaborate advisement [Tennyson *et al*. 1985], and linear control of the learning task [Goetzfried and Hannafin 1985]).

To assess the effectiveness and efficiency of LC strategies relative to PC strategies, Lee and Lee examined the previous studies on the comparison of LC with PC [Lee and Lee 1991]. They focused on two learning phases: knowledge acquisition and knowledge review, in conjunction with learners' generic metacognitive skills and previous domain knowledge. The results indicate that the effectiveness and efficiency of the LC strategy cannot be taken for granted. Learners' specific knowledge base affords a more reliable source for learning management decision and action than the generic type of metacognitive skills. For both knowledge acquisition and review, LC is superior to PC for the low level of domain knowledge, whereas this difference diminishes as the level of domain knowledge increases. It means LC strategy may work better for tasks of simple content structure, in which minimal domain knowledge is required.

Young [Young 1996] also compared outcomes of learners with high or low metacognitive skills in either LC or PC. The experimental results showed that the learners with low metacognitive skills learned less in LC than the learners in PC, whereas the learners with high metacognitive skills learned more in both LC and PC than the learners with low metacognitive skills.

Other similar studies have compared LC with PC through complex evaluation work on experimental results. In summary, the appropriate application of learner control strategies largely depends on learners' metacognitive skills, content structure, and the quality of system-provided lessons. The learner must exactly know *when* and *how* to control the learning progress. A learner with high metacognitive skills and greater prerequisite knowledge may do better with LC. But a poorly designed lesson with LC is far worse than a good lesson with PC.

Although there are varieties of controversies on learner control in software tutoring systems, it is obvious from the learners' learning processes and the society's needs that the application of learner control strategies in software tutoring systems is not only important but also necessary. In the earlier studies, learner control strategies were mostly used to improve the efficiency of learners' learning. With the emergence of web-based learning and interactive learning, and the increasing amount of adults accepting skills training in companies, the pure program control strategy cannot predict each learner's needs precisely, utilize learner's own resources maximally, or stimulate learners' motivations effectively. Especially in the context of microworlds – small modeling environments, learner control strategies have been not only more worthy but also necessary.

2.3. Learner Control Options

Although the term learner control is generally used, the actual control strategies vary. In general, there are four options regarding learner control:

- *Content and content sequencing*. Learners can select instructional goals and contents, and control the display order of the courses, topics, and screens within a lesson.
- *Pacing*. Learners can control the time spent on each lesson. Except for short video or audio sequences, learners can allocate different time periods to lessons for mastery according to their own needs.
- *Access to learning support*. Learners can control when to access which instructional components of lessons such as helps, examples, practice items, and coaches.
- *Instructional interaction method*. Learners can decide whether to learn individually or collaboratively, select peers as learning partners, and negotiate with systems on instructional goals and course contents.

In [Clark and Mayer 2002], only the first three options are given. The option of instructional interaction method is generated with the emergence of collaborative learning and constructive learning. All these control options represent the different tutoring and learning needs. In software tutoring systems, they can be controlled by learners to different degrees through corresponding software mechanisms. Note that although learners can control tutoring and learning, the learner control strategies are eventually implemented by software tutoring systems.

3. Software Tutoring Systems

The software tutoring systems are generally specific to situations where learners and tutors no longer have to meet like in the traditional classroom-based instruction. Similar to the definition of e-learning [Clark and Mayer 2002], a software tutoring system can be defined as a teaching and learning environment built on computers by way of CD-ROM, Internet, or Intranet with the following features: (1) include content relevant to the learning objective; (2) use instructional methods such as examples and practice to help learning; (3) use media elements such as words and pictures to deliver the content and methods; and (4) build new knowledge and skills to achieve individual learning goals or improve organizational performance.

The courses in software tutoring systems include both content (i.e., information) and instructional methods (i.e., techniques) helping people learn the content. They are digitized so they can be stored in electronic form, and delivered via computer using words in the form of spoken or printed text and pictures such as illustrations, photos, animation, or video. The purpose is to help individuals achieve educational goals or to help organizations build skills linked to improved job performance through training. The development of software tutoring systems has experienced three generations in rough and a variety of learner control strategies have been applied into each generation in different measures.

3.1. Computer-Assisted Instruction Systems

During the 1960s, educators were under considerable pressure to improve the quality of education as well as to educate a broader segment of society [Steinberg 1977]. Holt deplored the limitations of the widespread used lockstep instruction [Holt 1967]. Rapidly expanding community colleges had to accommodate increasingly numerous heterogeneous students. Teaching alternatives were sought at all levels and individualized instruction became a likely alternative. The application of computers to education made individualization even more feasible in terms of computer-assisted instruction (CAI, or computer-aided instruction). CAI systems were the first generation software tutoring systems. A CAI system presents a page of text or graphics, and puts up a different page depending upon the student’s answer [Hefferman and Koedinger 2002]. Although the implementation approaches of CAI vary significantly, the typical architecture can be briefly shown as in Figure 2, in which the computer plays the role of the tutor and communicates with the learner through human-computer interface in the form of text, graphics, multimedia, etc.

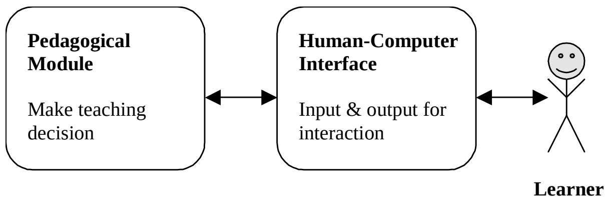

Figure 2. Computer-Assisted Instruction System

3.2. Model-Tracing Intelligent Tutoring Systems

Intelligent tutoring systems (ITS) are the earliest applications of artificial intelligence (AI) in education. Although ITS differ in a variety of ways, most have the characteristic structure outlined in Figure 3 [McArthur *et al*. 1993, Kay 2001].

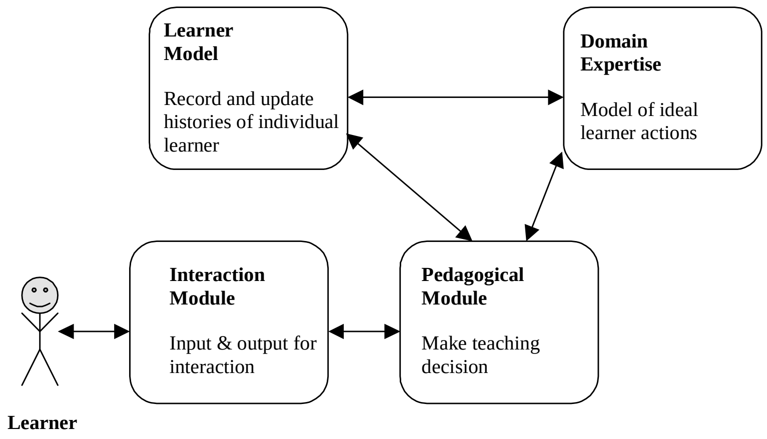
Learner Model
Record and update histories of individual learner
Domain Expertise
Model of ideal learner actions
Interaction Module
Input & output for interaction
Pedagogical Module
Make teaching decision
Learner

Figure 3. Intelligent Tutoring System

The learner interacts with the tutoring system via the *interaction module*, which may support natural language interaction or offer graphical interfaces. The *learner model* represents static beliefs about the learner and may stimulate the learner's reasoning, which is refined during interaction between learner and system. The *pedagogical module* determines the system's teaching actions, which draws upon the current state of the learner model and the domain expertise to teach individual learners. The *domain expertise* module enables the tutoring system to perform as an expert, which may be accessible by the learner in different ways such as glossary. The domain expertise module forms the heart of an ITS. The expertise system embeds sufficient knowledge of a particular topic area to provide "ideal" answers to questions, correct not only in the final result but in each of the smaller intermediate reasoning steps. The expertise system can also generate many different answer paths or goal structures. The detailed data structures generated in modeling expert reasoning permit ITS to explain their reasoning to learners at arbitrarily detailed levels [McArthur *et al*. 1993].

Some earlier ITS are model-tracing ITS. They are the second generation of software tutoring systems that allow the tutor to follow the line of reasoning of the learner [Hefferman and Koedinger 2002]. Like CAI systems before them, model-tracing ITS have attempted to implement traditional methods of learning and teaching exemplified by the one-on-one human tutoring interaction. In general, both CAI systems and model-tracing ITS are characterized as high *tutor control* and *short-answer* task format. Students learn by working a series of relatively brief questions, and the system plays the role of the task expert, controlling the selection of tutoring content, while the student responsible for answering them. The main differences between model-tracing ITS and CAI systems reflect engineering and psychological enhancements that permit model-tracing ITS to tutor in a *knowledge-based* fashion, enabling model-tracing ITS to teach in a much more detailed way than CAI systems.

3.3. Inquiry-Based Intelligent Tutoring Systems

Although CAI systems and model-tracing ITS permit learner control, most of interactions between learners and systems are tightly controlled by the software. The system selects the next task or problem, decides when the learner needs support and feedback in problem solving, and determines the information the learners receive. The learners can tailor information, for example, they may request more detailed explanations, but their behaviors are usually highly limited. Essentially, these systems focus on knowledge acquisition and knowledge review more than on knowledge construction, and are employing computer technologies to implement traditional tutor-centered teaching and learning such as one-on-one and drill-and-practice methods.

Another teaching and learning method, *inquiry-based* method (also described as student-centered, constructionist, constructivist, or discovery-based), has been deeply explored over the recent years and a transition towards inquiry-based instruction has begun. In [McArthur *et al*. 1993], systems that implement inquiry-based learning are collectively referred as *interactive learning environments* (ILEs). The ILEs have the following common characteristics:

- *Construction*. Students learn by constructing their own knowledge, not through lecture or organized drill-and-practice.
- *Learner control*. The tutor is viewed only as a guide.
- *Individualization determined by the learner*. Learners receive unique feedback and information as a function of their interaction with the environment built into the system instead of the tutor.
- *Rich feedback generated by the learner's interaction with the learning environment.* Fine-grained feedback arises from the learner's choices and actions rather than from discourse of the tutor.

Model-tracing ITS generally lack a good sense of interaction between learners and systems that may lie in the human tutoring as a dialog format. Based on model-tracing ITS and the features of ILEs, the third generation software tutoring systems have emerged which help learners to construct knowledge and tutor metacognitive processes. These inquiry-based ITS engage in *dialogs* with learners using multiple strategies. A learner's problem-solving behavior is the objective while the model tracing is the tool. The main difference between the first two generations and the third generation software tutoring systems is that the former is based on an "*instructionist*" approach to learning while the latter on a "*constructionist*" or "*constructivist*" approach to learning.

The classification of different generations of tutoring systems is not very strict. In fact, they have developed nearly in parallel. With the introduction of more complex tutoring and learning environments such as *collaborative learning* – a structured instructional interaction method among two or more learners to achieve a learning goal or complete an assignment [Panitz 1996], and *agent-based* tutoring systems – an agent is a computer system situated in some environment and capable of autonomous actions in this environment in order to meet its design objectives [Wooldridge 1999] which is often used in distributed tutoring systems, some researchers also classify the development of software tutoring systems into two generations, *solo-learning* (in form of individual) and *collaborative learning* (in form of small-group), from the perspective of learning. Thus the classification is becoming overlapped and obscured. Nevertheless, the most obvious fact is that more control is being given to the learner instead of the tutor in these systems. Therefore, the following discussion will focus on the learner control strategies in generic software tutoring systems.

3.4. Application of Learner Control Strategies in Software Tutoring Systems

The use of learner control strategies in software tutoring systems makes possible of the implementation of learner control. Corresponding to varieties of learner control strategies, the degree of learner control varies. A small control permits the learner to select to learn one topic before another, to invoke a help tool when uncertain about how to proceed, or to request extra examples of a concept being studied. A large measure of learner control permits the learner to construct his/her own problems, to interject spontaneous questions in the instructional dialogue, or to form learning collaboration with other learners. The following examples show the use of learner control strategies in software tutoring systems.

TICCIT (Time-shared Interactive Computer Controlled Information Television) is a milestone in the development of learner control strategy in software tutoring systems [Niemiec *et al*. 1996, Merrill 1980]. TICCIT was designed and developed by Merrill and his colleagues to teach “higher-order” concepts. It used learner controlled content selection, sequencing, self-pacing, and other instructional features. Together with the software support, special control keys on the keyboard allowed learners to access or change difficulty levels, numbers of examples and practice items, and the nature of advice. TICCIT also postulated four levels of conscious processing to support learner control: content selection, display selection, conscious cognition, and metacognition.

In MAIS (Minnesota Adaptive Instructional System), Tennyson and his associates [Tennyson 1980] used Bayesian statistical method to adjust the system’s instructional support on selecting the appropriate amount and sequence of instruction when learners learning concepts. Learners thus have the meaningful advisement on which to make judgments on content selection, content sequencing, and examples selection.

SCHOLAR is the first intelligent tutoring system [Duchastel 1986, Carbonell 1970], which was developed by Carbonell and colleagues. It incorporated a knowledge base storing the geographic facts of South America and was programmed to ask the student questions regarding these facts, as well as respond to the student’s own spontaneous questions. The natural language interface: *mixed-initiative dialogue*, in which either learner or system could ask a wide range of questions and expect responses from the other, gave control to the learner: the learner could interrupt the conversation at any point and reorient the interaction by asking the system a question.

WEST [Burton and Brown 1982] is a game environment which used the attraction of games to teach arithmetical and logical strategies to schoolchildren. In WEST, the learner progresses through the game according to his/her own strategies. Depending on the nature of the move and the context, the intelligent coach may interrupt the learner with strategic advice on better moves. The system is thus responsive to learner-initiated moves.

The Center for Interdisciplinary Research on Constructive Learning Environments (CIRCLE) in Carnegie Mellon University is devoting to build the third generation software tutoring systems. Geometry Explanation Tutor [Aleven *et al*. 2001], and Ms. Lindquist system [Hefferman and Koedinger 2002] are two examples. They are able to model-trace the students’ actions, knowledge-trace the students’ skills, provide mixed-initiative natural language dialogue as well as embedded subdialogs for follow-up questions, support the learner-controlled help-seeking behaviors, and request for self-explanation from learners to improve learners' metacognitive skills.

IDEAL (Intelligent Distributed Environment for Active Learning) is a web-based multiagent tutoring system [Shang *et al.* 2001]. It supports collaborative learning to expand learners' learning expertise, takes advantage of the power of learner-centered interaction, and encourages knowledge construction. Learners using this system can also control different types of agents for help.

This brief review covers only some of the applications of learner control strategies in software tutoring systems. While all these systems teach different subject matter in different ways, the distinguished feature of these systems from other traditional systems is the extent to which the learner controls the direction of the interaction and the progress of learning.

4. Typical Learner Control Strategies in Tutoring and Learning

From the perspective of application objective, learner control strategies should be able to maximize learner engagement, self-direction and self-evaluation, and to improve the learner's ability of the instructional strategy's management. From the perspective of specific implementation, learner control strategies should allow learners to control instructional events, sequence of these events, pace at which they wish them to happen, and whether or not they want to review the instruction from the summary practice session.

4.1. Types of Learner Control Strategies

There are varieties of learner control strategies in software tutoring systems. In general, they can be classified into the following six types although there are some overlaps:

- *Selection of instructional goal and content.* E.g., select or set learning topics and goals, select the difficulty degree of content, select content.
- *Sequencing and pacing of instructional materials and units.* E.g., content sequencing, self-pacing such as speed or time control of content presentation.
- *Selection and control of instructional components.* E.g., control the presentation of instructional components step-by-step: rule-example-practice, select the numbers of examples and practice exercises, select *worked examples* for highly structured topics, select high-order rules, select and use learning tools (e.g., charts, digital library, *virtual coaches* and *virtual companions*), control when to seek help, control contextual and presentational items: screen design and text density.
- *Metacognition control.* E.g., adjust based on *advisement*, *self-regulated learning*, control strategic planning such as teaching strategies and review strategies, revise through past errors, self-pacing, control feedback.
- *Learner construction.* E.g., self-explanation of learner to self or to tutor or to other learners, negotiation on instruction goals and objectives between learner and tutor, self-analysis and evaluation to learner self.
- *Selection and control of instructional interaction method.* E.g., select whether individual learning or collaborative learning, decide the size of collaborative group, select learning partners (via e-mails or chats or message boards), task allocation among partners, select coordination and communication tools, mixed-initiative dialogue between learner and tutor, or between learning partners.

A *worked example* is a step-by-step demonstration of how to solve a problem or perform a task [Clark and Mayer 2002]. It can reduce mental work as one of the most powerful methods for learners to build new and rich knowledge in long-term memory. Learners often choose worked examples over verbal descriptions to help them complete problem assignments.

A *virtual coach* is an onscreen pedagogical agent provided by the system during the learning process [Cassell *et al*. 2000]. It plays the role of a tutor but not a human tutor. The virtual coach can help guide the learning process through step-by-step explanation of how to solve each problem, and improve communication by engaging and motivating learners. The learner can select a teaching strategy from different types of virtual coaches. A *virtual companion* [Kay 2001] is a learner agent provided by the system during the learning process. It is a simulated learner able to learn together with the real learner. A learner can have virtual companions during learning.

Advisement is a process in which learners are given advice as to what actions they should take in a lesson, based on the program's evaluation of their responses to lesson exercises [Tennyson and Buttrey 1980, Clark and Mayer 2002]. It is a variation of adaptive control that leaves learner control in place. The advisement information consists of *diagnosis* (the learner's initial level of knowledge compared with the desired learning criterion) and *prescription* (the amount of sequence of instruction necessary for the learner to obtain the objective) generated and continuously updated by the system. Rather than automatically branching to appropriate sections of the instruction as in adaptive control, in the learner control with advisement, the system provides recommendations to the learner regarding what to select in the next step, so that the learner can adjust the learning process or ignore the advice.

Self-regulated learning refers to learners' systematic use of metacognitive, motivational, and behavioral strategies to achieve academic goals [Zimmerman 1990]. Strictly speaking, it should not be included into the learner control strategies here since it is a relatively high-level concept. Learners' self-regulatory skills can help to stimulate internal motivations within learner control instruction, which is very important for the learners in the distance learning environment where there are no tutors to monitor their activities and status.

These strategies types are ordered by the difficulty degree and the time of emergence. The classification is primarily based on different control options presented in Section 2.3. For example, the first two types correspond to the options: content and content sequencing, and pacing. The third type corresponds to the option: access to learning support. The last two options correspond to the option: instructional interaction method.

Merrill postulated four levels of conscious processing specific to TICCIT system (see Section 3.4), and classified learner control strategies in form of these levels. This classification was done based on the different types of events happened to the learner: external events and internal events. Merrill only listed limited strategies in [Merrill 1980]. Similarly, Clark and Mayer classified the learner control strategies in form of control options (see Section 2.3) but they also only presented very few specific strategies. Niemiec *et al.* presented more learner control strategies but they did not classify those strategies [Niemiec *et al*. 1996]. Moreover, all of them ignored the learner control strategies with respect to learner construction and instructional interaction, which emerge from the new generation of software tutoring systems. So the classification of learner control strategies here is relatively complete.

4.2. Relationship Between Strategies and Educational Theories

Learner control strategies are built on numerous educational theories. The relationship between the previously described strategies and the educational theories is listed in Table 1. There exist overlaps among these theories and one learner control strategy may be based on multiple theories. Thus only the main theories as well as the corresponding relationships are listed.

Table 1. Overview of Learner Control Strategies

Educational Theories	Application Basis	Learner Control Strategies
Experimental learning	Addresses the needs and wants of the learner. Self-initiated learning is the most lasting and pervasive.	Select or set learning topics and goals
Levels of processing	The greater the processing of information during learning, the more it will be retained and remembered.	Content sequencing Control the presentation of instructional components step-by-step: rule-example-practice Select the numbers of examples and practice exercises
Component display theory	Classifies learning along two dimensions: content (facts, concepts, procedures, and principles) and performance (remem-bering, using, and generalities). Specifies four primary presentation forms: rules, examples, recall, and practice.	Select instructional components: content and presentation strategies
Structural learning theory	Identifies the rules to be learned for a given topic and breaks them into atomic com-ponents. Teaches the simplest solution path for a problem first and then teaches more complex paths until the entire rule has been mastered.	Select high-order rules
Cognitive load theory	Reduces working memory load during the learning process in order to facilitate the changes in long-term memory associated with schema acquisition.	Control contextual and presentational items: screen design and text density Select the difficulty degree of content Select worked examples for highly structured topics Select and use learning tools
Metacognition theory	Learner's cognitive management skills, including setting goals, monitoring progress, and adjusting strategies as needed. Essentially involves self-knowledge, planning, reflection and review.	Select or set learning goals Self-analysis and evaluation to learner self Control when to seek help Adjust based on advisement Self-regulated learning Select teaching strategies or review strategies Revise through past errors Self-pacing Control feedback

Constructivist learning	Learning is an active process in which learners construct new ideas or concepts based upon their current/past knowledge. The learner selects and transforms information, constructs hypotheses, and makes decisions, relying on a cognitive structure to do so.	Self-explanation of learner to self or to tutor or to other learners Negotiation on instruction goals and objectives between learner and tutor Self-analysis and evaluation to learner self
Situated learning Social development theory Collaborative learning	Learning and full cognitive development requires social interaction and collaboration. Collaborative learning encourages active student participation in the small-group learning process.	Select instructional interaction method: individual learning or collaborative learning Decide the size of collaborative group Select learning partners Task allocation among partners Select coordination and communication tools Use virtual companions
Conversation theory	Learning occurs through conversations about a subject matter that serves to make knowledge explicit.	Mixed-initiative dialogue between learner and tutor, or between learning partners

5. Learner Control Strategies Implementation

Learner control strategies in software tutoring systems are eventually implemented through software design and development. Due to the wider application of computer technology in education, more and more software mechanisms are involved in learner control strategies. In this section, the related software mechanisms will be presented from different perspectives.

5.1. System Architecture

The original software tutoring systems are individual programs installed on personal computers or single servers accessible from multiple terminals. The learner control strategies in these systems are generally limited within selection of instructional goal and content, sequencing and pacing of instructional materials, and partial control of instructional components.

The system architecture of modern software tutoring systems is more complex. The most common is the client-server structure as well as the three-tier architecture – a second-tier server (or an "agent") is introduced between the client and the server. These systems are generally web-based, agent-based, and distributed systems. The pedagogical part and knowledge base as well as varieties of learning tools may be placed on different servers. Each learner may access those servers from the client-end. Some specialized agents with different expertise may be used to manage the learner's personal profile or mediate communication and collaboration. The previously described system IDEAL is a typical example. IDEAL adopts three-tier architecture and is implemented using technologies of Internet, WWW, digital libraries, and software agents such as student agents, teaching agents and course agents.

Except for the traditional learner control strategies, multiple educational theories (e.g., constructivist learning theory, social development theory, and cognitive theory) are applied to provide opportunities for learners to engage in interactive, creative, and collaborative activities such as knowledge construction, group learning, and usage of virtual coaches as well as other learning tools. Based on the web-based architecture, the learner control strategies can be implemented using some prevalent development tools such as XML, HTML, and JAVA.

5.2. Human-Computer Interaction

The most direct strategies supporting learner control exist in the human-computer interaction part. The earliest software tutoring systems only provided the command-line interaction method, so the learner control was very limited and inconvenient. Now, the most common interaction method between learners and systems is WIMP interface (Window, Icon, Menu, and Pointing). Most software tutoring systems with GUI use WIMP interface to implement learner control. WIMP is composed of a series of GUI tools such as window, icon, menu, button, and mouse. Learners can use these interaction tools to select and sequence contents, select and control instructional components, select and use other system-provided tools. The major parts of the navigational system supporting web-based learning also depend upon WIMP interface.

5.2.1. Design Principles of Context and Presentation

In [Clark and Mayer 2002], three principles of the human-computer interface design are recommended: multimedia principle, contiguity principle, and modality principle.

- *Multimedia principle.* Both words (printed text and spoken text) and graphics are presented for learners' selection rather than words alone. The relevant graphics are helpful for learners' deeper cognitive processing and selection of course content.
- *Contiguity principle.* On-screen words are placed near the graphics to which they refer and the linked information does not cover related information on the primary screen. The feedback appears on the same screen as the question and the practice of including exercise directions also appears on the same screen on which the actions are to be taken. These screen designs help learners select and control instructional components.
- *Modality principle.* Whenever the graphic or animation is the focus of the words and both are presented simultaneously, the words are spoken instead of being printed. The learner should be able to select the presentation format of words.

These interface design principles are based on cognitive theory especially the cognitive load theory. They should be considered into the design of software tutoring systems.

5.2.2. Navigational Features to Implement Learner Control

In web-based learning, guided tours, course menus, links, and buttons (e.g., forward, backward, and exit) are the most common navigational techniques supporting learner control [Clark and Mayer 2002].

Guided tours are overviews accessible from the main menu screen that demonstrate the resources available in a course. The learner can sequence contents with guided tours within a course.

Course menus list all the lessons and topics. Learners can select specific lessons and topics within a course. To help the learner track progress within a course, it is common to highlight lessons or topics that have been completed.

Links are placed within the primary teaching frame. Learners can access content from other sites on the Internet, or from other sections within the course such as the rules, examples and practice exercises. Links that take the learner off the teaching screen should be used sparingly. According to the contiguity principle, the high use of links in a lesson is negatively correlated to learning due to cognitive overload [Niederhauser *et al*. 2000]. Moreover, to access important instructional course elements such as practice exercises, default navigation options should be designed to lead to them; thus learners are required to make a deliberate choice to bypass practice [Schnackenberg and Sullivan 2000].

Buttons that activate forward, backward, and exit options permit learners to sequence and self-pace contents within the lesson. For important course elements, after the user presses the exit button, a dialogue box should be pop up to get his/her confirmation.

5.3. Conceptual Architecture

Modern software tutoring systems are commonly intelligent tutoring systems with the four-part architecture previously illustrated in Figure 3: interaction module, domain expertise, pedagogical module, and learner model. In fact the boundaries between these elements are not always clear. With the employment of collaborative learning and agent technologies in software tutoring systems, the conceptual architecture of a software tutoring system has become more complex. For example, the interaction module expands to also provide interaction between learners as well as between learners and agents. Figure 4 depicts a high-level view of the emerging class of software tutoring systems with learner control strategies and evolves from [Kay 2001].

Figure 4 focuses on demonstrating the system-provided support to learner control, thus the typical four models illustrated in Figure 3 are hidden. Figure 4 lists five types of tools possibly provided by software tutoring systems: domain tools, generic tools, virtual coaches, virtual companions, and real companions. When the learner has a rich collection of tools available, one important level of learner control is the choice of *whether* or not and *how much* to use each tool.

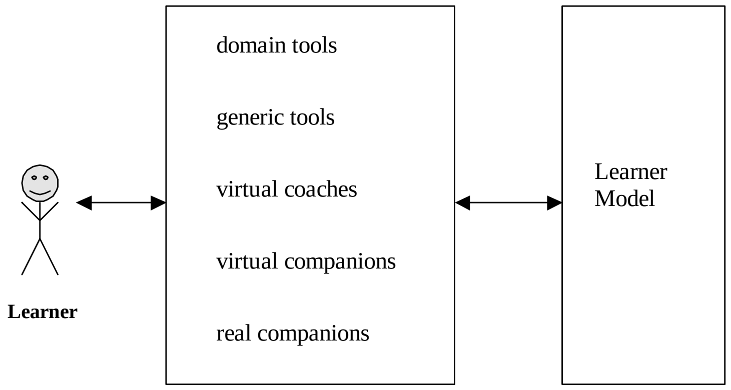

Figure 4. Conceptual Architecture Supporting Learner Control

The *domain tools* are useful for the learning domain knowledge and may include a simulation environment that enables the learner to explore ideas or practice important and time-critical tasks like medical learning domain. The *generic tools* include text editors, charts, graphical tools, and so on. Learners can selectively use these tools to reduce cognitive load. The *virtual coaches* provide the learner with the choice of teaching strategies. They can appear as different roles like a planner or an assessor. The *virtual companions* and *real companions* are tools especially provided for collaborative learning. Learners can use these agents tools to select different types of learning companions, form cooperative groups, allocate tasks or share resources among group members, track the learning progress, and manage their achievement records and versions of solutions. Companions can assist each other and a learner can be a coach when he/she trains another learner on how to use new technological tools in a collaborative project.

Ultimate learner control requires the learner be able to see aspects of the underlying system control. Since the learner model lies at the heart of individualization, an open learner model is a fundamental part of learner control. In an ITS, the system utilizes the learner model to adapt to learners' needs and aptitudes. In Figure 4, the learner model is an independent entity. A learner can access and control personal profile information via provided tools, also assess correctness of the model. The learner's access to the model can improve his/her appreciation of the system's learning goal, comprehension of the complex system, and metacognitive skills by supporting reflection and planning. All of which are helpful for the implementation of learner control. In addition, programmer accountability may be enhanced as he/she knows the learner will have access to the model.

Such a conceptual architecture can support learner construction, inquiry-based interaction, and collaborative learning. It complies with the cognitive learning theory.

5.4. Intelligent Techniques

Three typical AI features are used mostly in modern software tutoring systems: knowledge base, natural language interface, and intelligent agents. These features have significantly influenced the implementation of learner control in software tutoring systems.

5.4.1. Knowledge Base

The design of software tutoring systems requires that intuitive notions about learning held by system designers be abstracted out of their experience and formalized [Kearsley 1985]. This principle is implicitly involved with gradual accrual of knowledge through particular interactions with a knowledge structure. The knowledge base of an ITS is a structure in which each element of knowledge is defined in terms of other elements in the structure. According to [Duchastel 1986], the intelligence of an ITS is due in great part to its knowledge of contents. In terms of learner control, a knowledge base enables the learner to interact more directly with knowledge not just with one representation of knowledge (like symbol strings in a frame in CAI). The structure of an ITS knowledge base permits spontaneous access to its diverse knowledge elements, which enables the interaction between learner and system not confined within the sequential access necessitated by traditional CAI systems.

5.4.2. Natural Language Interface

Traditional CAI systems are built in the relatively deterministic instructional design paradigm [Duchastel 1986]. These systems deal with learner input by partitioning information into small frames such that the learner's response to the question terminating the frame can generally be anticipated and therefore handled in a pre-determined branching manner. This feature of CAI in turn necessitates high system control, and makes the system mistreat learners' unanticipated requests as unrecognized responses. The application of learner control strategies is thus limited. Modern ITS are characterized by a preference for non-deterministic instructional strategies (the idea of reactive learning), embodying the need for learner control. The natural language interface enables the system to analyze a wide range of learner input, and enables learner control in the form of interruption and reorientation of the conversation. This is essentially due to the grounding of natural language processing grammars in the knowledge base of the system.

In software tutoring systems, natural language processing is involved in multiple learner control strategies regarding knowledge construction, learner-tutor conversation (e.g., self-explanation, or negotiation on instructional goals and objectives), and learner-learner conversation (e.g., negotiation on task allocation during collaborative learning). Aleven *et al*. [Aleven and Koedinger 2000, Aleven *et al*. 2001] applied self-explanation, an effective learner control strategy, to their third generation tutoring systems, PACT Geometry Tutor and Geometry Explanation Tutor. The support for self-explanation in the PACT Geometry Tutor is limited: learners did not explain in their own words, selected explanations from menus, and provided very abbreviated explanations. The Geometry Explanation Tutor was created by adding natural language understanding (or dialogue) capabilities to the PACT Geometry Tutor. The system engages learners in a restricted form of dialogue in order to help them state general explanations that justify their problem solving steps. The systems' natural language understanding components are based on a knowledge base with ontology and explanation hierarchy, and also use a statistical text classifier as a backup. This classifier is based on the Naïve Bayes classification method.

5.4.3. Intelligent Agents

Intelligent agents originate from Distributed AI technology. An intelligent agent is capable of *flexible* autonomous actions in order to meet its design objectives [Wooldridge 1999]. Intelligent agents impose islands of tutor-control in a relatively large expanse of learner controlled inquiry activities. The agents' interventions are carefully chosen so that learner initiative is not interrupted unless there is very strong evidence that the learner is thrashing in unprofitable ways. An agent does not need to have a complete or fully accurate cognitive model of the learner's misconception, so the learner is free to disregard the advice offered by the agents and simply proceed with his/her plan [McArthur *et al*. 1993].

IDEAL [Shang *et al*. 2001] consists of a number of specialized intelligent agents with different expertise: student agents, teaching agents, and course agents, that were used to manage students' personal profiles, serve as intelligent tutors of courses, and mediate communication among students. Learners have options to choose whether or not to use an agent and take its advice. WHAT (Web-based Haskell Adaptive Tutor, [Lopez *et al*. 2002]) is another intelligent tutoring system designed to help students learn the programming language Haskell. It supports collaborative learning. In addition to real students, intelligent agents can be added to the study groups as virtual companions, which may not be informed to the real students. The only difference between virtual companions and real companions is that virtual companions do not need to use the web interface. With the development of agent and intelligent agent technology, intelligent agents are having more applications in software tutoring systems.

5.5. Classification of Learner Control Strategies in Software Mechanism

No research work has been found yet on the classification of learner control strategies from the perspective of software mechanism. Based on the previous description to the main aspects of a software tutoring system, the learner control strategies in software tutoring systems can be broadly classified into three types from the perspective of software mechanism:

- *Use of system tools.* E.g., domain tools, generic tools, agents, collaboration tools, and tools for learner model access.
- *Learning status control.* E.g., access the learner model, assess the current progress, and choose pending tasks willing to work on.
- *Internal Intelligence support.* E.g., natural language processing, knowledge base, and selective use of intelligent agents.

6. Open Problems and Proposed Solutions

The learner control strategies in software tutoring systems are involved with multiple areas like education, psychology, and computer science. There are still many open problems such as how to expand learners' knowledge source, improve the system's advisement, and combine learners' metacognitive skills with learner control.

6.1. Expand Learners' Knowledge Source

In software tutoring systems, digital library can be used as a component of the system to provide learners with related domain knowledge (e.g., [Shang *et al*. 2001]). Some systems also provide low-cost, on-line help in the form a glossary (e.g., [Aleven and Koedinger 2000]). Learners specify the requirement, while the system is responsible for finding specified knowledge and presenting to learners. Even if the knowledge source and the system software are located on different servers, the system can still execute this task. However, if the system does not provide such a knowledge source, or during learning the learner needs information unavailable in the system, the problem of information access occurs.

One easy approach is that the system permits the learner to leave the current content temporarily and surf on the Internet for the knowledge acquisition. But this method is time-consuming and may be invalid especially when the learner is a novice in the searched field and thus cannot filter information efficiently.

The proposed solution here is to use *search spiders (or search agents)* to obtain knowledge from remote places for the learner. A search spider is a software process capable of roaming on the Internet, gathering relevant information on behalf of its owner, and coming back home having performed the duties [Chen *et al*. 1998]. The use of search spiders can alleviate learner's burden, save learning time, increase the accuracy of querying, and decrease the communication traffic. The system can preset some search spiders for learners' selection. Each search spider should be as simple as possible so that a learner only needs to provide primary clues. When a search spider receives a query task, it will go to other sites, filter irrelevant information, and return with the objective information. The system even may provide the experienced learners with the autonomous programming chance to add simple code segments into the tutoring system to customize their own search spiders.

6.2. Improve Advisement

The effective application of learner control strategies in software tutoring systems not only depends on the software mechanism support, but also on learners' individual characteristics such as cognitive capabilities, metacognitive skills, and motivations. The advisement provision of the system is an effective strategy to help learner control.

Presently, the software tutoring system provides advisement mainly based on the comparison between the learner's current learning progress and the desired learning criterion. It does not take into consideration the learner's characteristics. Thus such advisement is not individualized and cannot match each learner's needs effectively.

One approach to improving advisement is to collect the learners' important characteristic information as much as possible, analyze the collected information, and estimate learners' behavior tendencies. Then the system can give more pertinent advisement.

To collect the learners' characteristic information in a greater degree, *multi-modality interface* can be employed through multimedia equipments. When a learner is interacting with the system, for example, taking an oral self-explanation with natural language, the audio and video equipments available in the system will capture the learner's face expression and tone variance. The captured signals can be transformed into system-recognizable information in expression and phonics recognition techniques. Then the system combines the collected characteristic information with the learner's learning records to check the learner's calibration accuracy level [Stone 2000], self-confidence degree, learning speed, and other features via specific scaling techniques. Based on the analysis results, the system can estimate learners' behavior tendencies such as rashness in answering questions, low self-evaluation, and slow reading speed. When the system gives advisement, it will take these characteristic factors into consideration, and advise learners to avoid poor learner control.

6.3. Combine Learners' Metacognitive Skills with Learner Control

The learner's metacognitive skills play a significant role in learner control. The design of a good software tutoring system should take into consideration learners' metacognitive skills. For the high metacognitive learners, more learner control strategies are helpful to motivate their creativities; whereas for the low metacognitive learners, less learner control strategies should be provided to avoid important contents being bypassed. So a good software tutoring system should be able to change available learner control strategies to adapt to learners' metacognitive skills. However, it is difficult for the software tutoring system to exactly determine each learner's metacognition capability and decide which learner control strategies provided.

Presently, the educators have numerous methods to test the degree of learners' metacognitive skills. In general, a series of questions as well as the corresponding scaling policy are designed to evaluate the learners' answers. It can differentiate the levels of learners' metacognitive skills. This artificial way lacks flexibility and objectiveness as a course may last months and a learner's metacognitive skills may vary during this period. If each time when the learner works with the system, the system supposes his/her metacognitive skills exactly like the original scaling result, the learner's control degree will be affected.

One solution is to equip the tutoring systems with intelligent measurement agents to judge learners' metacognitive skills dynamically. The system sets built-in measurement agents with systematic questions and the scaling policy. Periodically, when a learner begins to learn in the system, the agent asks the learner to take the test, measures his/her metacognitive skills, and classifies the skills through combining the current measurement results with the past ones. Then the system is able to dynamically decide which learner control options provided for the learner. Thus high metacognitive learners can have more control options while low metacognitive learners less control options.

7. Conclusions

Learner control is often beneficial especially for learners with effective learning abilities in the interactive learning environments. The critical question is not whether to use learner control but how to use it in software tutoring systems. Presently, the commonly used learner control strategies concentrate on the selection of topics, sequencing and pacing of contents, selection of instructional components, and control over learner-tutor and learner-learner interaction.

The effective application of learner control strategies in software tutoring systems depends not only upon software mechanisms but also upon educational theories. The individuals' learning abilities significantly influence the use of learner control strategies. The design of a good software tutoring system should take into consideration learners' cognitive abilities, metacognitive abilities, and other factors involved in educational psychology. For high-ability learners, more learner control can be applied, while for low-ability learners, less learner control should be used. Thus one of duties of software tutoring systems is to teach learners how to learn, which is similar to the function of a human tutor. Otherwise, even numerous learner control strategies are useless.

Different learner control strategies are generally used in different situations, so it is difficult to differentiate which strategy is better than others. The learning outcome can be used as an effective criterion of the quality measurement to learner control strategies. But the implementation cost of learner control strategies in software tutoring systems should also be taken into consideration deliberately.

Intelligent agent techniques have been and will be used in software tutoring systems more in the future. This tendency is very obvious and natural from the perspective of labor division in human society. Agents can replace learners to do some work, but how to efficiently utilize agents is the learner's responsibility. Thus the use of agents should be simplified and easy to understand.

References

[Aleven and Koedinger 2000] Aleven, V. and Koedinger, K. R. The Need for Tutorial Dialog to Support Self-Explanation. In *Building Dialogue Systems for Tutorial Applications*, eds. Rose, C. P. and Freedman, R., Papers from the *2000 AAAI Fall Symposium*, pages 65-73, Menlo Park, CA: AAAI Press, 2000.

[Aleven *et al*. 2001] Aleven, V., Popescu, O., and Koedinger, K. R. Towards Tutorial Dialog to Support Self-Explanation: Adding Natural Language Understanding to a Cognitive Tutor. In *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*, eds. Moore, J. D., Redfield, C. L., and Johnson, W. L., *Proceedings of AI-ED 2001*, pages 246-255, Amsterdam: IOS Press, 2001.

[Burton and Brown 1982] Burton, R. and Brown, J. S. An Investigation of Computer Coaching for Informal Learning Activities. In *Intelligent Tutoring Systems*, eds. Sleeman, D. and Brown, J. S., pages: 79-98, Academic Press, New York, 1982.

[Carbonell 1970] Carbonell, J. AI in CAI: An Artificial Intelligence Approach to Computer Assisted Instruction. *IEEE Transactions on Man-Machine Systems*, 11(4): 190-202, 1970.

[Cassell *et al*. 2000] Cassell, J., Sullivan, J., Prevost, S., and Churchill, E. *Embodied Conversational Agents*. Cambrige, MA: MIT Press, 2000.

[Chen *et al*. 1998] Chen, H., Chung, Y. M., Ramsey, M, and Yang, C. C. An Intelligent Personal Spider (Agent) for Dynamic Internet/Intranet Searching. *Decision Support Systems*, 23(1): 41-58, 1998.

[Clark and Mayer 2002] Clark, R. C. and Mayer, R. E. *e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning*. Jossy-Bass Publishers, 2002.

[Duchastel 1986] Duchastel, P. Intelligent Computer Assisted Instruction Systems: The Nature of Learner Control. *Journal of Educational Computing Research*, 2(3): 379-393, 1986.

[Forbus and Feltovich 2001] Forbus, K. D. and Feltovich, P. J. The Coming Revolution in Educational Technology. In *Smart Machines in Education*, eds. Forbus, K. D. and Feltovich, P. J., pages 3-5, AAAI Press/MIT Press, 2001.

[Goetzfried and Hannafin 1985] Goetzfried, L. and Hannafin, M. J. The Effect of the Locus of CAI Control Strategies on the Learning of Mathematical Rules. *American Educational Research Journal*, 22: 273-278, 1985.

[Gray 1987] Gray, S. H. The Effect of Sequence Control on Computer Assisted Learning. *Journal of Computer-Based Instruction*, 14: 54-56, 1987.

[Hefferman and Koedinger 2002] Hefferman, N. T. and Koedinger, K. R. An Intelligent Tutoring System Incorporating a Model of an Experienced Human Tutor. In *Proceedings of the 6th International Conference on Intelligent Tutoring Systems (ITS 2002)*, pages 596-608, Biarritz, France and San Sebastian, Spain, June 2-7, 2002.

[Holt 1967] Holt, J. *How Children Fail*. New York: Pitman Publishing, 1967.

[Kay 2001] Kay, J. Learner Control. *User Modeling and User-Adapted Interaction*, 11:111-127, 2001.

[Kearsley 1985] Kearsley, G. Microcomputer Software: Design and Development Principles. *Journal of Educational Computing Research*, 1: 209-220, 1985.

[Lee and Lee 1991] Lee, S. and Lee, Y. H. K. Effects of Learner-Control Versus Program-Control Strategies on Computer-Aided Learning of Chemistry Problems: For Acquisition of Review? *Journal of Educational Psychology*, 83(4): 491-498, 1991.

[Lopez *et al*. 2002] Lopez, N., Nunez, M, Rodriguez, I., and Rubio, F. Including Malicious Agents into a Collaborative Learning Environment. In *Proceedings of the 6th International Conference on Intelligent Tutoring Systems (ITS 2002)*, pages 51-60, Biarritz, France and San Sebastian, Spain, June 2-7, 2002.

[Mager 1961] Mager, R. On the Sequencing of Instructional Content. *Psychological Reports*, 9: 405-413, 1961.

[McArthur *et al*. 1993] McArthur, D., Lewis, M., and Bishay, M. The Roles of Artificial Intelligence in Education: Current Progress and Future Prospects. RAND DRU-472-NSF, http://www.rand.org/hot/mcarthur/Papers/role.html, 1993.

[Merrill 1980] Merrill, M. D. Learner Control in Computer Based Learning. *Computers and Education*, 4: 77-95, 1980.

[Newman 1957] Newman, S. Student vs. Instructor Design of Study Method. *Journal of Educational Psychology*, 48: 328-333, 1957.

[Niederhauser *et al*. 2000] Niederhauser, D. S., Reynolds, R. E., Salmen, D. J., and Skolmoski, P. The Influence of Cognitive Load on Learning from Hypertext. *Journal of Educational Computing Research*, 23: 237-255, 2000.

[Niemiec *et al*. 1996] Niemiec, R. P., Sikorski, C., and Walberg, H. J. Learner-Control Effects: A Review of Reviews and a Meta-Analysis. *Journal of Educational Computing Research*, 15(2): 157-174, 1996.

[Panitz 1996] Panitz, T. A Definition of Collaborative vs Cooperative Learning. http://www.lgu.ac.uk/deliberations/collab.learning/panitz2.html, 1996.

[Park and Tennyson 1980] Park, O. C. and Tennyson, R. D. Adaptive Design Strategies for Selecting Number and Presentation Order of Examples in Coordinate Concept Acquisition. *Journal of Educational Psychology*, 72: 362-370, 1980.

[Ross *et al*. 1980] Ross, S. M., Rakow, E. A., and Bush, A. J. Instructional Adaptation for Self-Managed Learning Systems. *Journal of Educational Psychology*, 72: 312-320, 1980.

[Schnackenberg and Sullivan 2000] Schnackenberg, H. L. and Sullivan, H. J. Learner Control Over Full and Lean Computer-Based Instruction Under Differing Ability Levels. *Educational Technology Research and Development*, 48: 19-35, 2000.

[Shang *et al*. 2001] Shang, Y., Shi, H., and Chen, S. An Intelligent Distributed Environment for Active Learning. *Journal of Educational Resources in Computing*, 1(2): 308-315, 2001.

[Steinberg 1977] Steinberg, E. R. Review of Student Control in Computer-Assisted Instruction. *Journal of Computer-Based Instruction*, 3(3): 84-90, 1977.

[Steinberg 1989] Steinberg, E. R. Cognition and Learner Control: A Literature Review, 1977-1988. *Journal of Computer-Based Instruction*, 16(4): 117-121, 1989.

[Stone 2000] Stone, N. J. Exploring the Relationship between Calibration and Self-Regulated Learning. *Educational Psychology Review*, 12(4): 437-475, 2000.

[Tennyson 1980] Tennyson, R. D. Instructional Control Strategies and Content Structure as Design Variables in Concept Acquisition Using Computer-Based Instruction. *Journal of Educational Psychology*, 72(4): 525-532, 1980.

[Tennyson and Buttrey 1980] Tennyson, R. D. and Buttrey, T. Advisement and Management Strategies as Design Variables in Computer-Assisted Instruction. *Educational Communication and Technology Journal*, 28(3): 169-176, 1980.

[Tennyson *et al*. 1985] Tennyson, R. D., Park, O. C., and Christensen, D. L. Adaptive Control of Learning Time and Content Sequence in Concept Learning Using Computer-Based Instruction. *Journal of Educational Psychology*, 77: 481-491, 1985.

[Wooldridge 1999] Wooldridge, M. Intelligent Agents. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, ed. Weiss, G., pages 27-77, The MIT Press, 1999.

[Young 1996] Young, J. D. The Effect of Self-Regulated Learning Strategies on Performance in Learner Controlled Computer-Based Instruction. *Educational Technology Research and Development*, 44: 17-27, 1997.

[Zimmerman 1990] Zimmerman, B. Self-Regulated Learning and Academic Achievement: An Overview. *Educational Psychologist*, 25(1): 3-17, 1990.